Tema 2
Rețele de calculatoare
Offline messenger

Ștefana Biceada, Grupa 2E2

5 Decembrie, 2022

1 Introducere

Proiectul Offline Messenger constă in dezvoltarea unei aplicații, care permite schimbul de mesaje între utilizatori. Aplicația este implementată folosind un server și un client și are următoarele funcționalitați: ultilizatorii care sunt conectați pot comunica între ei, utilizatorii offline vor primi mesajele atunci cand se vor conecta, pentru fiecare utilizator se poate afișa și istoricul conversațiilor, utilizatorii pot răspunde specific la anumite mesaje (funcția de reply) și posibilitatea de deconectare (logout).

2 Tehnologii de utilizare

- Implementarea Server/client este bazată pe protocolul TCP(Transmission Control Protocol), deoarece acesta este un serviciu care asigură livrarea ordonata a unui flux de octeți de la un program la altul, precum si confirmarea primirii datelor. De asemenea,in cadrul TCP-ului mașinile care trimit si cele care primesc sunt conectate si comunică una cu cealaltă tot timpul. TCP asigură ca datele să ajungă la destinatar corect, fără erori.
- Comunicarea dintre server si client este realizată prin socket-uri pentru a se trimite in mod sigur fluxul de mesaje. Socketurile ofera posibilitatea de a trimite informații între procese atat local cat și între procese ce de află pe mașini diferite.
- Pentru gestionarea proceselor am ales thread-uri pentru ca acestea oferă o performanță eficientă, deoarece thread-urile execută porțiuni de cod in paralel. De asemenea, acestea simplifică structura programului și asigură concurența.
- Pentru stocarea informațiilor (conversațiile, numele utilizatorilor, parolele acestora, etc.) am folosit fișisere simple deoarece, funcționalitatea oferită de acestea este potrivita pentru problemă.

- Programul este implementant in C/C++.

3 Arhitectura aplicației

Aplicația ,,Offline messenger" oferă următoarele comenzi pe care utilizatorii le pot folosi:

- Login() Utilizatorii se pot autentifica la aplicație.
- Register() Utilizatorii se pot înregistra in aplicație.
- See Users() Afișează lista utilizatorilor.
- Send message() Clienții pot trimite mesaje altor utilizatori care se află in lista de useri.
- Reply() Utilizatorii pot răspune in mod specific mesajelor primite.
- Show message while offline() Utilizatorii pot vedea mesajele primite in perioda in care aceștia nu erau conectați la server.
- User history conversation() Utilizatorii pot vedea intregul istoric al conversațiilor.
- Logout() Utilizatorii se pot deconecta de la aplicație.

4 Detalii de implementare

Un scenariu pozitiv in care aplicația funcționeaza corect poate fi acesta: doi utilizatori deja înregistrați in baza de date/xml file se conectează la aplicație, folosind comenzile disponibile, scrie corect.

Un scenariu negativ, care va crea probleme poate fi: scrierea greșita a comenzilor, in acest caz aplicația va semnala comanda greșita. De asemenea, un utilizator poate introduce un username care nu este valid (username folosit deje sau care contine simboluri simboluri care nu sunt acceptate), caz in care va primi o eroare.

La deschiderea aplicației, utilizatorul se poate loga, inregistra sau poate ieși din aplicație. Pentru a se loga in aplicatie, utilizatorul trebuie sa introduca username-ul sau precum si parola, daca acestea sunt găsite in fișier, utilizatorul este logat, in caz contrar, un mesaj ce indrumă clientul spre a se inregistra. Dupa autentificare, utilizatorul are acces la funcționalitațile principale ale aplicației. Acesta poate conversa cu alți uilizatori care sunt online sau poate scrie mesaje utilizatorilor offline, aceștia vor putea vedea mesajele odata cu prima logare in aplicație. Aceste notificări sunt afișate dintr-un fișier, iar dupa ce sunt printate pe ecran, conținutul fișierului este șters. Clienții pot vedea istoricul conversațiilor, acestea fiind afișate dintr-un fișier, identificat cu ajutorul id-ului userului.

- Diagrama urătoare ilustrează arhitectura aplicației:

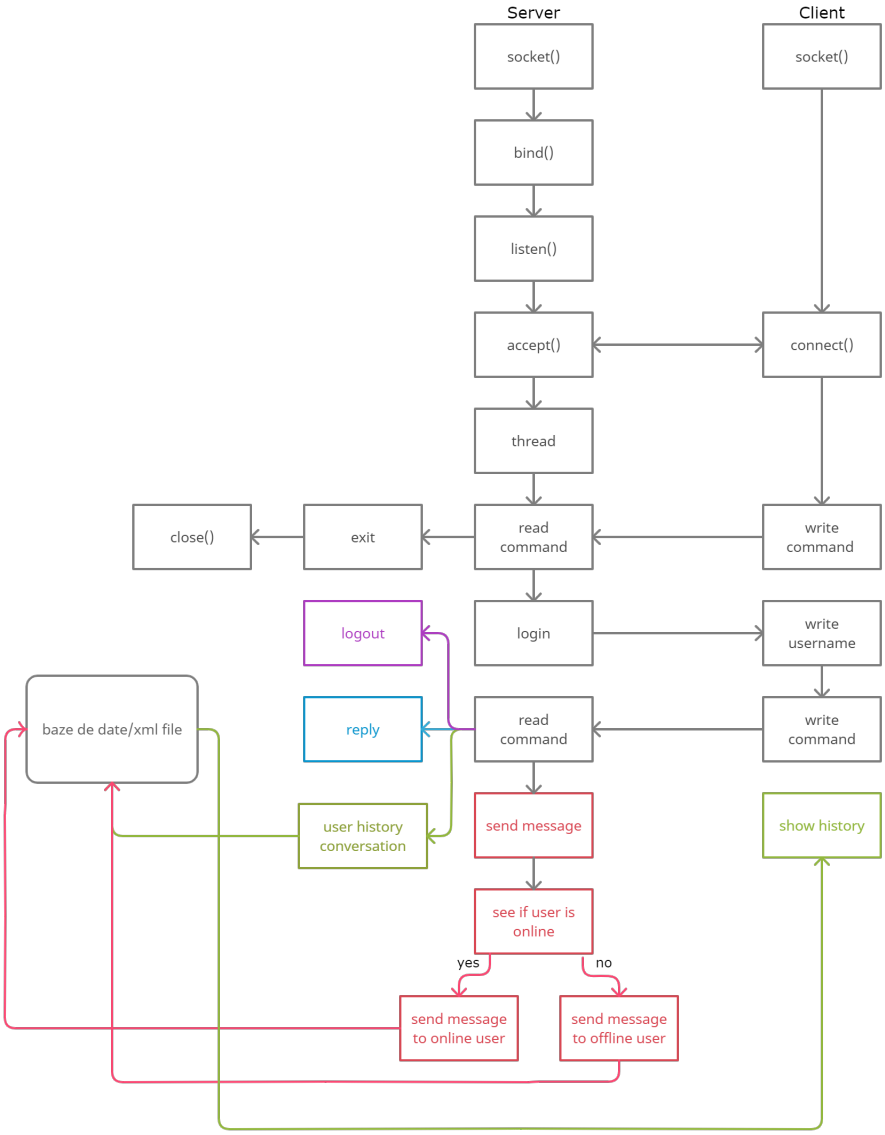


Fig.1 Crearea socketului.

```c
/* crearea unui socket */
if ((sd = socket (AF_INET, SOCK_STREAM, 0)) == -1)
    {
        perror ("[server]Eroare la socket().\n");
        return errno;
    }

    /* utilizarea optiunii SO_REUSEADDR */
int on=1;
setsockopt(sd,SOL_SOCKET,SO_REUSEADDR,&on,sizeof(on));

/* pregatirea structurilor de date */
bzero (&server, sizeof (server));
bzero (&from, sizeof (from));

/* umplem structura folosita de server */
/* stabilirea familiei de socket-uri */
    server.sin_family = AF_INET;
/* acceptam orice adresa */
    server.sin_addr.s_addr = htonl (INADDR_ANY);
/* utilizam un port utilizator */
    server.sin_port = htons (PORT);

/* atasam socketul */
if (bind (sd, (struct sockaddr *) &server, sizeof (struct sockaddr)) == -1)
    {
        perror ("[server]Eroare la bind().\n");
        return errno;
    }
```

Fig.2 Threadurile folosite pentru a servi clientii

```c
typedef struct thData{
    int idThread; //id-ul thread-ului tinut in evidenta de acest program
    int cl; //descriptorul intors de accept
}thData;
```

Fig.3 Meniul aplicației după logare

```
29
30  int menu2;
31  void print_menu2()
32  {
33      printf("%s\n", "Select command from: ");
34      printf("%s\n", "[1] See users");
35      printf("%s\n", "[2] Send message to user");
36      printf("%s\n", "[3] History with user");
37      printf("%s\n", "[4] Own history");
38      printf("%s\n", "[5] Menu");
39      printf("%s\n", "[6] Logout");
40  }
41
```

Dupa logare daca clientul doreste sa vorbeasca cu un alt utilizator acesta poate selecta comanda nr 2. Serverul primeste comanda de la client și apeleaza functia SendMessage, unde se face distincția dintre utilizatorii online si cei offline.

Fig.4 Funcția SendMessage, ce face trimiterea spre funcțiile pentru trimiterea mesajelor.

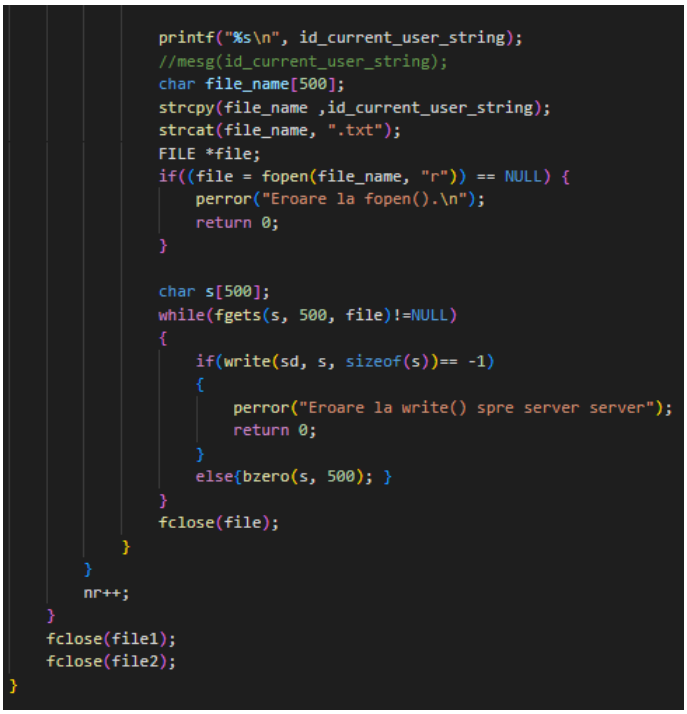
```
int SendMessage(int sd, thData data)
{
    int is_online = 0;

    //CITIM USERUL CU CARE VREM SA VORBIM DE LA CLIENT
    char id[50];
    bzero(id, sizeof(id));
    if (read (sd, id,sizeof(id)) < 0){
        perror ("Eroare la read() de la client.\n");
    }
    printf("User selected: %s\n", id);

    //CAUTAM USERUL IN LISTA DE UTLIZATORI ONLINE
    int j;
    for(int i = 0; i <= nrClients; i++){
        if(clients[i]->id_user == atoi(id) && clients[data.idThread]->id_user != atoi(id)){
            is_online = 1;
            j = i;
        }
    }

    if(is_online == 1){
        sendMessageOnline(sd, data, atoi(id), j);
    }
    else{
        sendMessageOffline(sd, data, atoi(id));
    }
}
```

Functia pentru afisarea mesajelor intre utilizatorii online funcționeaza pe baza printării conținutului fișierului ce conține conversația dintre cei doi clienți. Cand fucnția este apelată, sunt create sau deschise fișierele corespunzătoare, urmand ca severul sa primească mesajul de indrud ce către client. Acesta este scris in fișiere, apoi conținutul fișierului este afișat in client. Funcția reply foate

fi apelată prin inceperea mesajului cu caracterul ”-”.

Fig.5 & 6 Funcția SendMessageOnline, ce permite convorbirea intre doi utilizatori.

```
while(chatting == 0){
    printf("%s\n", "Waiting for a message from client.");

    //CITIM MESAJUL DE LA CLIENT
    bzero(message, sizeof(message));
    if (read (sd, message,sizeof(message)) < 0){
        perror ("Eroare la read() de la client.\n");
    }
    printf("Mesajul primit: %s\n", message);
    //SCRIEM IN FISIERUL PERSONAL AL USERULUI

    if(message[0] == '#'){
        chatting = 1;
        printf("%s\n","Clientul doreste sa termine conversatia");
    }else{
        if(message != NULL){
            if(message[0] != ' '){
                OwnFile(message, sd, data);
                if(message[0] == '-'){
                    fprintf(file1, "nr.%d: ", nr);
                    fprintf(file2, "nr.%d: ", nr);

                    fprintf(file1, "%d replies to ", clients[data.idThread]->id_user);
                    fprintf(file2, "%d replies to ", clients[data.idThread]->id_user);
                }else{
                    fprintf(file1, "nr.%d: ", nr);
                    fprintf(file2, "nr.%d: ", nr);

                    fprintf(file1, "%d: ", clients[data.idThread]->id_user);
                    fprintf(file2, "%d: ", clients[data.idThread]->id_user);
                }
                fprintf(file1, "%s\n", message);
                fprintf(file2, "%s\n", message);
                fflush(file1); fflush(file2);
                }
                //fclose(file1); fclose(file2);

                //TRIMITEM CLIENTULUI NR DE LINII
                int nr2 = countlines(id_current_user_string);
                char nr_linii2[10];
                sprintf(nr_linii2, "%d", nr2);
                if (write (sd, nr_linii2, 10) <= 0){
                    perror ("[client]Eroare la write() spre server.\n");
                    return errno;
                }
```



```
                printf("%s\n", id_current_user_string);
                //mesg(id_current_user_string);
                char file_name[500];
                strcpy(file_name ,id_current_user_string);
                strcat(file_name, ".txt");
                FILE *file;
                if((file = fopen(file_name, "r")) == NULL) {
                    perror("Eroare la fopen().\n");
                    return 0;
                }

                char s[500];
                while(fgets(s, 500, file)!=NULL)
                {
                    if(write(sd, s, sizeof(s))== -1)
                    {
                        perror("Eroare la write() spre server server");
                        return 0;
                    }
                    else{bzero(s, 500); }
                }
                fclose(file);
            }
        }
        nr++;
    }
    fclose(file1);
    fclose(file2);
}
```

Daca utilizatorul cu care clientul logat dorește sa vorbeasca este offline atunci funcția SendMessageOffline este apelată. In acesta funcție se primește mesajul de la client, fiind scris intr-un fișier corespunzător userului offline, urand ca la prima sa logare conținutul acestui fișier sa fie printat.

Fig.7 Funcția SendMessageOnline, ce permite scrierea mesajelor unui user offline.

```c
int sendMessageOffline(int sd, thData data, int nr_id)
{
    printf("%s\n", "User offline");

    if (write (sd, "User offline!", 50) <= 0){
        perror ("[client]Eroare la write() spre server.\n");
        return errno;
    }

    char message[500];
    int chatting = 0;
    while(chatting == 0){
        printf("%s\n", "Waiting for a message from client.");

         if(strncmp(message, "#", 1) == 0) {
            chatting = 1;
        }else{

            bzero(message, sizeof(message));
            if (read (sd, message,sizeof(message)) < 0){
                perror ("Eroare la read() de la client.\n");
            }

            Notificari(message, sd, data, nr_id);
        }
    }
}
```

5 Concluzii

Aplicația poate fi îmbunătățită prin folosirea unei baze de date sau fișierelor xml, deoarece acestea ofera o gestionare mai bună a informațiilor. De asemenea, o interfața grafică simplă ar putea reprezenta un factor ce face utilizarea aplicației mai placută.

6 Bibliografie

[1] https://profs.info.uaic.ro/~computernetworks/cursullaboratorul.php

[2] https://en.wikipedia.org/wiki/Transmission_Control_Protocol